CMU - SCS
15-415/15-615 Database Applications
Spring 2013, C. Faloutsos
Homework 7: Database Application
Released: Tuesday, 03/26/2013
Deadline for PHASE 1: **Tuesday, 04/02/2013**
Deadline for PHASE 2: **Thursday, 04/11/2013**

Reminders - IMPORTANT:

- Like all homework, it has to be done **individually**.
- Please **typeset** your answers.
- This assignment has two phases:
  Phase I is due 4/2 as hard copy in class.
  Phase II is due 4/11 as both hard copy in class and electronically.
- We will hold recitations on the following Wednesdays, : 3/27, 4/3, and 4/10, 2:30-3:30pm.

Reminders - FYI:

- Weight: 30% of homework grade.
- The points of this homework add up to 100.
- Rough time estimates: **20 - 30 hours**. Start early!

Q1. Introduction

You are to create a movie recommendation web site, CMUFlix. The goal is to learn how to set up a web site with a database back-end. This assignment is divided into two phases, as we describe in the Roussopoulos and Yeh methodology (see later for exact pointers). In the first phase, you will plan your implementation strategy at a high level, and submit documentation that describes the major design decisions you have made. The first phase permits the TAs to evaluate your progress

at an early stage and provide feedback. The second phase is the implementation of the system. For both phases you are expected to give extensive documentation for the system. The target audience of the document is the fictitious person who will have to maintain the system afterwards.

Q2. Requirements

Q2..1 Data Requirements

The system allows users to login, "like" a movie, and get movie recommendations.

1. **User data**: For every user, we want to record the login name (e.g. "etaralova415"), the password, and the email address. Login names should be unique. The password cannot be empty.
2. **Movie data**: we will provide a very small movie database - there is only one key, `mtitle`, the movie title.
3. **Movie "likes"**: for each user, the system should store which movies they have "liked." A user cannot "unlike" a movie. If a user "likes" a movie multiple times, only one count is recorded.

Q2..2 Functionality Requirements

T1. Registration (already provided for you - see below): Before a user can start using your system, he/she needs to register by providing login name, email address, and choosing a password. If a login name chosen by a user during registration already exists, the system should give an error message and prompt for a different login. You can just store the password as clear text, and you need not verify the validity of the email address. Note: a sample registration is already provided for your reference (`www/register.jsp`, see the README[1] file, which contains all the detailed setup instructions). Please make any modification to meet the above requirement.

T2. Login/Logout: A user should be able to login with the login/password he/she provided during registration. For security, no one should be able to view any page of your website without logging in. Also, once logged in, the user should not need to log in to view other pages. A login session is valid until the user explicitly logs out, or the user remains idle for 30 minutes.

Hint: In this assignment we use Tomcat, which provides easy session and cookie management primitives using cookies and server-side persistence. The default timeout for the sessions for your tomcat JSP server is 30 minutes, so you need not modify that. Please refer to Chapter 6, Section 6 and 7 of Thinking in Enterprise Java (see below) for detailed example of using session and cookies in JSP.

[1] `http://www.cs.cmu.edu/~christos/courses/dbms.S13/hws/HW7/README`

T3. Profile Page: When the user logs in, he/she should see his/her "profile information" including (1) all the information about this user (login name, email address), (2) the list of movies they have "liked."

T4. "Like" a movie: a user should be able to navigate to a web page which lists all movies in the database with a checkbox "Like" next to each entry they have not yet "liked."

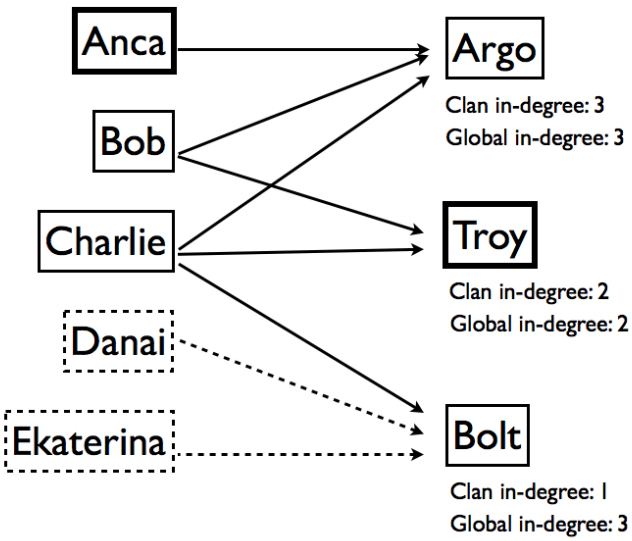

Figure 1: Recommend a movie example (T5), and "clan": the logged in user is Anca; her "movie clan" is Bob and Charlie (since they have at least one movie in common with Anca); the system should recommend 'Troy' and 'Bolt', with 'Troy' first, because it is more favorite within Anca's "clan".

T5. Recommend a movie: a user should be able to navigate to a web page which generates up to 5 movie recommendations for this user, based on their existing "likes."

- If the user hasn't "liked" anything before, display the top 5 movies with largest in-degree (= most "likes"). Break ties by the alpha order on the title.
- If a user has "liked" at least one movie, then recommend movies that similar-minded people "liked", sorted by popularity (among those similar-minded people). See Figure 1 for an example: to make a recommendation for user "Anca" (in double box), find his/her "movie clan", i.e., persons who liked at least one movie in common with "Anca" (that is,

Bob and Charlie, her 'movie clan'); then, report the most popular movies within Anca's "clan". That is, display up to 5 movies with largest in-degree from their "movie clan," in decreasing order, breaking ties by decreasing alpha order of the title. In the example of the figure, the movies to recommend to Anca would be 'Troy' (clan-in-degree = 2), and 'Bolt' (clan-in-degree = 1). (Of course, we don't want to return 'Argo', since Anca has liked it already.)

T6. Reporting: The system should print summary of statistics of user activity in a report page. The page should contain:

- T6..1 The total number of registered users in the system.
- T6..2 The total number of "likes" in the system from all users.
- T6..3 The list of movies that no one has liked, sorted in alpha order on the title.
- T6..4 The list of the "avid users" - the usernames of users with the most number of "likes," limit to top 10, sorted by the number of likes. Break times by alpha order of the username.

Q3. Setup

A web server (Tomcat 6) with JSP support has already been set up for you. To access it, you should log into your account on newcastle.db.cs.cmu.edu (the one you used for HW2 and HW 6). Please clean up and back up your directory. The detailed instructions can be found here: `http://www.cs.cmu.edu/~christos/courses/dbms.S13/hws/HW7/README`

Q4. Recommended Code Organization

In this project, you will write HTML, JSP, and/or Java code. The www directory contains all the code accessible online. Here is the recommendation for your code organization (# for comments). Feel free to deviate from it, but following it will make grading easier.

```
~ / # home directory
   /src # source code directory, including Java, JSP and HTML
   /www # where you deploy your web application
      /WEB-INF
        web.xml # your web configuration file, see Tomcat doc
        classes  # put all compiled Java classes (.class files) here
        lib  # any external library, e.g. postgresql-8.3-603.jdbc4.jar
      #no need to modify:
      /META-INF  # Tomcat specific context configuration directory
```

Note: In the handout sample implementation of registration, all processing is included in www/register.jsp. While this provides a simple solution, it may create complex JSP pages in a large project. A more elegant way is to use the MVC pattern - separating functionality of model, view and control. There are many such tools available, e.g. *struts*. You are welcome to use *struts* if you would like, but any working solution will get full points (with, or without *struts*).

Q5. Phases

The two phases of this assignment follow the work-processes methodology from Adaptable Methodology for Database Design by Roussopoulos and Yeh [IEEE Computer, May 1984] (`http://www.cs.cmu.edu/~christos/courses/dbms.S13/slides/CMU_ONLY/Roussopoulos-Yeh.pdf` , the lecture slides are also available at `http://www.cs.cmu.edu/~christos/courses/dbms.S13/slides/20methodology.pdf`).

[Phase I] Environment and Requirement Analysis, System Analysis and Specication, Conceptual Modeling, Task Emulation.

[Phase II] Implementation and Testing.

Q6. Point Distribution and Deliverables

Phase I: report in hard copy, due 4/2 1:30pm in class [35 points]

The Phase I report should contain the following:

1. The top-level information flow diagram, (very important - also, don't forget the system boundary). [1 pt]
2. The list of documents. [1 pt]
3. The document forms, including the assumptions and design decisions you made. [3 pts]
4. The E-R model (omit attributes, to avoid cluttering the diagram). Make sure you specify cardinalities of the relationships following the format in hw1 solution. [5 pts]
5. List of the attributes for each entity and relationship. [2 pts]
6. Explanations of the any non-obvious entities and relationships. [3 pts]
7. The schema in the relational model. Make sure the schema is in a good form (BCNF or 3NF). [5 pts]
8. Explanations (e.g., primary keys, additional functional dependencies, explanation why a table is not in BCNF etc.) [2 pts]
9. The SQL DDL statements to create the above relational schema. [2 pts]
10. The SQL DML statements for all the tasks. [10 pts]
11. Run the registration task given in the demo code. Report the output generated by the webpage when you register your andrew-id. [1 pt]

Phase II: report in hard copy, due 4/11 1:30pm in class [15 points]
The Phase II report should contain the documentation produced in this phase:

1. The source program listing - print out your code. [5 pts]
2. Usually, for such a system we would require the user's manual for the system - how to install and setup. However, since this is trivial in this case, we are skipping this step. [0 pts]
3. Your testing efforts: erroneous cases that your system can detect and handle reasonably (e.g., non-member trying to access the system, etc.). You are expected to create several users and several scenarios to test the system. [10 pts]

Phase II: report in electronic files, due 4/12, 1:30pm, electronically [50 points]
You should copy all of your source code into the directory hw7_your_andrew_id415.

The weights of the tasks are as shown in Table 1. Notice that about two thirds of the points are for basic implementation that supports the desired functionality, and the rest are for error checking (as described above). Note: GUI is not the emphasis in this assignment.

Task	Basic Implementation	Testing/error checking
T1. Registration	0	2
T2. Login/logout	5	2
T3. Profile page	7	3
T4. Like	7	3
T5. Get recommendation	7	3
T6. Report	8	3
Total	34	16

Q7. Optional Documentation and Resources

Thinking in Enterprise Java (`http://www.mindviewinc.com/downloads/TIEJv1.1.zip`). Chapters 4-6 cover how to connect to databases, servlets and JSP. Chapter 3 on RMI is optional.

Thinking in Java (`http://www.mindviewinc.com/downloads/TIJ-3rd-edition4.0.zip`): a good introductory book on Java programming;

Thinking in Patterns (`http://www.mindviewinc.com/downloads/TIPatterns-0.9.zip`): Java design principles and methods.

Tomcat 6.0 manual is available at `http://tomcat.apache.org/tomcat-6.0-doc/index.html`.

struts is a framework for building enterprise web application. It uses MVC pattern (Model-View-Controller). The API is available at http://struts.apache.org and you could find some nice tutorial at `http://struts.apache.org/2.x/docs/tutorials.html`.